Top-Down Structurally-Constrained Neural Response Generation with Lexicalized Probabilistic Context-Free Grammar

Wenchao Du
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
wenchaod@cs.cmu.edu

Alan W Black
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
awb@cs.cmu.edu

Abstract

We consider neural language generation under a novel problem setting: generating the words of a sentence according to the order of their first appearance in its lexicalized PCFG parse tree, in a depth-first, left-to-right manner. Unlike previous tree-based language generation methods, our approach is both (i) top-down and (ii) explicitly generating syntactic structure at the same time. In addition, our method combines neural model with symbolic approach: word choice at each step is constrained by its predicted syntactic function. We applied our model to the task of dialog response generation, and found it significantly improves over sequence-to-sequence baseline, in terms of diversity and relevance. We also investigated the effect of lexicalization on language generation, and found that lexicalization schemes that give priority to content words have certain advantages over those focusing on dependency relations.

1 Introduction

Neural encoder-decoder architectures have shown promise and become very popular for natural language generation. Over the past few years, there has seen a surging interest in sequence-to-sequence learning for dialog response generation using neural encoder-decoder models (Vinyals and Le, 2015; Serban et al., 2017). Typically, an encoder encodes conversational context (source side) information into vector representations, and a decoder auto-regressively generates word tokens conditioned on the source vectors and previously generated words.

Two problems arise with the standard left-to-right decoding mechanism. First, no future information is available at any step of the decoding process, while the study of linguistic dependency structure shows that certain words depend on the others that come right to them. Second, preceding words define the context for following words in left-to-right, auto-regressive language models, while linguistic theories may prefer other hierarchies (e.g., adjectives modifying nouns, adverbs modifying verbs). Psycho-linguistics studies also suggest that human may first generate the abstract representation of the things to say, and then linearize them into sentences (Dell et al., 1999).

Therefore, it is appealing to consider language generation in alternative orders. This poses a greater challenge because a mechanism in extra to word generation is needed for deciding the position of each word. Some recent works adopt a syntax-free approach to address this problem. (Mehri and Sigal, 2018) proposed a middle-out decoder that starts from the middle of sentences and finishes the rest in forward and backward directions. (Mou et al., 2016) and (Li and Sun, 2018) start with one or two predicted keywords and generate the rest of sentences in a similar fashion. Others incorporate tree structures without syntactic relations and categories. (Zhou et al., 2018) canonicalizes the dependency structures of sentences into ternary trees, and generate only the words top-down. Yet another line of work aim to model the full syntactic trees. (Gū et al., 2018) generates phrase structures and part-of-speech tags along with words for machine translation. (Dyer et al., 2016) generates shift-reduce action sequences of context-free grammars in addition to words for language model and parsing. But words are still generated in left-to-right order in their approaches.

In the domain of dialog, we believe language generation can benefit from alternative orders, for the same reasons argued earlier. On the other hand, in human conversations, the structure of utterances usually correspond with dialog states (e.g., wh-noun or wh-adverb phrases are more

Proceedings of NAACL-HLT 2019, pages 3762–3771
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

likely to be used in a request state), so modelling phrase structures can potentially help capturing discourse level information. In order to be able to generate complete syntactic trees, while be flexible about word generation order at the same time, the use of lexicalized grammar becomes a natural choice.

2 Related Work

Recent years has seen works in language model and generation through alternative orders. (Zhang et al., 2016) developed a top-down neural architecture for language model that alternates between four LSTM decoders according to given dependency relations. (Ford et al., 2018) proposed a two-stage language model, of which the first stage is a language model that generates templates, and the second stage is a translation model that fills in the blanks. Word generation order varies with the choice of words that are generated at different stages.

Language generation with tree structures has been explored more thoroughly for neural machine translation. (Eriguchi et al., 2017) and (Aharoni and Goldberg, 2017) generate CFG trees in bracketed form. (Wu et al., 2017) generates the sequence of transitions to form dependency trees. More recent works have focused on explicitly generating tree structures (Wang et al., 2018; Gū et al., 2018).

Regarding neural architectures for tree generation in the field of natural language processing, (Dong and Lapata, 2016) and (Yin and Neubig, 2017) use a single decoder with parent-feeding mechanism to generate logical forms and programming codes. (Gū et al., 2018) applied the doubly-recurrent neural networks of (Alvarez-Melis and Jaakkola, 2016) with attention mechanism to machine translation. Their model uses two decoders, of which one memorizes the ancestors, and the other remembers the siblings. (Wang et al., 2018) also uses two decoders, but one for generating words and the other for generating syntax trees.

In the domain of dialog response generation, the use of syntactic structures is under-studied. (Mou et al., 2016) and (Li and Sun, 2018) considered starting with keywords, and finish the sentences in forward and backward directions. Their models in principle are not tree-structured. The closest thing to our knowledge is by (Zhou et al., 2018). They proposed to convert dependency trees to ternary trees, but ignore the type of dependency relations. In other words, they modelled on trees of which the nodes and edges have no labels. The key difference between their approach and ours is that we generate syntax trees with labels, and word choices are also constrained by the labels.

3 Design Choices

We first consider the following three requirements when generating an L-PCFG syntax tree:

Deciding the structure of children. Several mechanisms have been proposed for deciding the structure of children of each node in the context of tree generation. One of them decide tree topology by using a sequence model to generate children one by one and predict stopping tokens (Alvarez-Melis and Jaakkola, 2016). Then there is a simpler approach that treats each combination of the labels of the children as one token, and predict such tokens when generating the parent node (Yin and Neubig, 2017). For language generation, we adopt the second approach and predict the combination of the labels of children, i.e. the rules, for two reasons: (i) the space of grammar rules is generally sparse even when its dimensionality is exponential of the number of labels, and (ii) with sequential generation of labels, as in the first approach, it is hard to enforce the labels of the children to form a valid grammar rule.

Deciding the heir of a node. Recall the definition of lexicalized PCFG: let W, N, R be the sets of lexicons, labels, and rules, where each rule is one of the following forms:

- $X(h) \to h$
- $X(h) \to Y_1(h_1) \dots Y_k(h_k)$ such that there exists i, $h_i = h$.

where $X, Y_1, \dots, Y_k \in N$, $h, h_1, \dots, h_k \in W$. We do not restrict ourselves to Chomsky Normal Form, and rules can have any number of children. The i^{th} children in the second case is called the *heir*. The key difficulty to top-down generation of lexicalized PCFG parse tree is deciding which child would be the heir. One way is to make explicit decision to select the child by adding a switch variable, at the cost of increasing the complexity of the problem. Instead, we make a change to the second case above, and simplify the problem by restricting the rules to be of the following form:

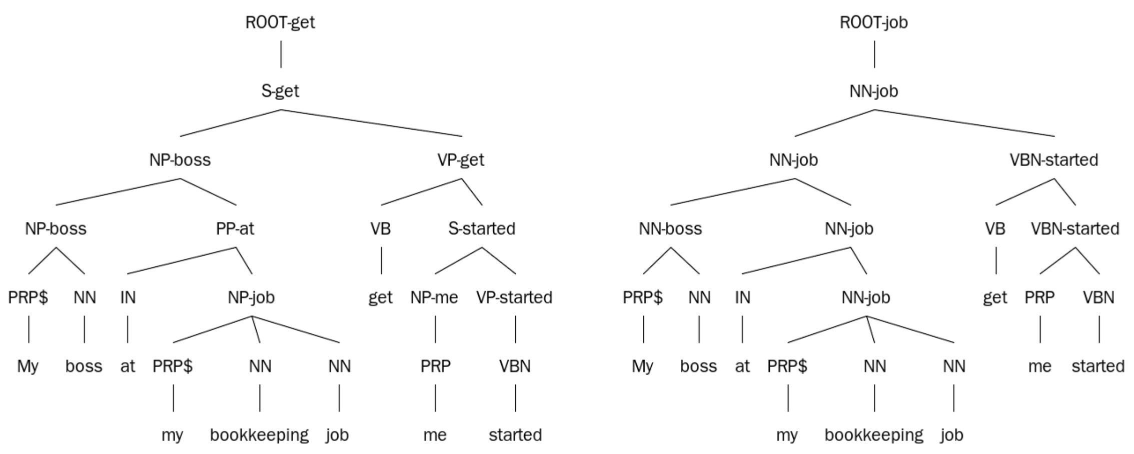

Figure 1: The first tree is the result of Stanford parser. The second one is obtained by performing content-based lexicalization on the first tree. All labels are replaced by part-of-speech tags of heirs. Unary rules NP-me → PRP and VP-started → VBN are removed.

- $X(h) \rightarrow Y_1(h_1)\dots Y_k(h_k)$ such that there exists i, $h_i = h$ and $Y_i = X$.

In other words, the heir would inherit both the lexicalization and the label of the parent (with the possible exception that the root node may produce children that are not labeled with "root"). When generating a parent node and its children, we restrict the choice of rules to those containing the label of the parent, so the heir can be inferred from the chosen rules by looking for the child that has the same label as its parent (in case there are multiple children that have the same label, we choose the rightmost one; other heuristics are possible). Note that under such restriction we end up with parse trees in which all labels are part-of-speech tags.

Sequentialization of a tree. Previous works adopt various construction orders of trees. (Zhang et al., 2016), (Zhou et al., 2018), (Alvarez-Melis and Jaakkola, 2016), and (Gū et al., 2018) generate trees through level-order traversal (breadth-first, left-to-right), whereas (Yin and Neubig, 2017) and (Wang et al., 2018) generate trees through pre-order traversal (depth-first, left-to-right). (Kuncoro et al., 2018) also experimented with bottom-up and left-corner construction orders for language model. While finding the optimal order of generating trees is beyond the scope of this work, we follow (Yin and Neubig, 2017) and (Wang et al., 2018), and generate lexicalized PCFG syntax trees through pre-order traversal.

4 Methods

4.1 Definitions

In this paper, we give the following graph-theoretic definition of L-PCFG syntax trees. Let W and N be the sets of lexicons and labels. Let $R = \bigcup_{k=1}^{\infty} N^k$ be the set of production rules. Then an L-PCFG syntax tree T is an *ordered tree* defined by the triple of vertices $v \in V \subset W \times N \times R \times \mathbb{N}$, edges $e \in E \subset V \times V$, and bijection $f : V \times \mathbb{N}^+ \rightarrow V$ such that $(v, f(v, j)) \in E$, where j range from 1 to the number of children of v. The fourth coordinate of v is the index of its heir, that is,

$$(w, n, r, i) = f((w_0, n_0, r_0, i_0), i_0)$$
$$\implies w = w_0, n_0 = n$$

We say a node $v = (w, n, r, i)$ is a *leaf* if r is unary:

$$\text{v is a leaf} \iff r \in N$$

The parent of v_k is denoted by $v_{p(k)} = (w_{p(k)}, m_{p(k)}, n_{p(k)}, i_{p(k)})$.

4.2 Generation Procedure

Following previous work, we sequentialize L-PCFG parse trees and generate its content in an auto-regressive manner. When generating k^{th} node, we predict the lexicalization w_k and the rule r_k. The pre-order history available when generating k^{th} node is $n_1 \cdots n_{k-1}, w_1 \cdots w_{k-1}, r_1 \cdots r_{k-1}$, denoted by

H_k. The label "ROOT" is given at the start of generation. The label of the k^{th} node is inferred from the production rule of its parent and the order of k^{th} node among its siblings, and is used as input together with H_k. When a leaf node is reached, the program backtraces until it finds an ancestor that has unfinished child, and proceeds to the first such child.

We factor the joint probability of w_k and r_k into two component: a word model and a syntax model, as follow:

$$P(w_k, r_k \,|n_k, H_k) = \\ P(w_k \mid n_k, H_k) \cdot P(r_k \mid n_k, H_k)$$

The details of both models are given in the following sections.

4.3 Lexicalization Schemes

We parse the responses in training corpus using the lexicalized parser by (Klein and Manning, 2003) (which we call Stanford parser for the rest of this paper). We then replace the label of each node with that of their heir in a bottom-up manner. Unary rules at non-leaf nodes are removed as they become redundant given our definition of lexiclaized PCFG.

Stanford parser lexicalizes PCFG phrase structures by looking for the most likely combination of phrase structure and dependency structure. While their approach is optimized for parsing, the syntax trees lexicalized this way has a drawback for the purpose of generation. Empirically, their parser tends to lexicalize the first few nodes with auxiliary verbs or common verbs (e.g. be, must), and in some cases prefer function words over content words (e.g. in preposition phrases). We hypothesize that choosing content words over functions, or infrequent words over frequent words as lexicalization heads will help making the generation more specific and meaningful. Hence, we consider two alternative lexicalization schemes:

Content-based lexicalization. We rank words according to their part-of-speech in the sentence by the following order: nouns $>$ verbs $=$ adjectives $>$ adverbs $>$ everything else. If two words have the same rank, we give priority to the rightmost one. See Figure 1 for an example.

Frequency-based lexicalization. We ignore part-of-speech information and rank all words by their frequencies. We regard less frequent words as more important.

4.4 Encoding Tree Histories

To represent the state of a tree node by encoding its pre-order history H_k, we use 3 LSTMs to memorize the lexical and grammatical contents in H_k.

Encoding lexicalization history. We use 2 LSTMs to encode the lexicalization history, i.e. $w_1 \cdots w_{k-1}$: a *surface* decoder, L_s, which takes the lexicalization of the leaves in the history as inputs; and an *ancestor* decoder, L_a, which is given the lexicalization of the ancestors of the current node. This is another form of doubly-recurrent neural networks. Different from (Alvarez-Melis and Jaakkola, 2016), we chose to encode leaves instead of siblings. Denote the lexicalizaiton of leaves and ancestors in H_k by $\{w_{l(k)_i}\}$ and $\{w_{a(k)_i}\}$. We show that for an L-PCFG syntax tree, $\{w_{l(k)_i}\}$ and $\{w_{a(k)_i}\}$ sufficiently cover the lexical content of H_k:

Proposition. For any index set $I_k \subset \{1, 2, \ldots, k-1\}$, if $w_n \in \{w_j\}_{j \in I_k}$ for all $n < k$, then $\{l(k)_i\} \bigcup \{a(k)_i\} \subset I_k$, i.e. $\{l(k)_i\}$ and $\{a(k)_i\}$ together is the minimal index set to cover $w_1 \cdots w_{k-1}$.

Encoding syntactic history. We encode the previous labels and rules using the full history with a single LSTM, L_g. At step k, the input to L_g is the concatenation of the embeddings of n_k, the rule of parent node $r_{p(k)}$, and depth of the node d. The depths of nodes deeper than 10 are rounded down to 10.

4.5 Encoding Source and Attention Mechanism

We adopt attention mechanism into our architecture for response generation. We use a one-layered LSTM to encode the dialog history, which is the concatenation of the last few utterances. The initial hidden states of L_s, L_a, and L_g is computed from the last hidden state of source encoder using 2 fully-connected layers with rectified linear activation. At time step t, the concatenation of the hidden states of L_s and L_a at step $t-1$ is used as the key for querying the source. The attention weights are the inner products of the key and the hidden states at source side, normalized by softmax function. The weighted sum of source hidden states results in the attention context, $c(k)$

4.6 Decoding

Denote the hidden states of L_s, L_a, L_g at node k as $h_s(k), h_a(k), h_g(k)$. Denote the softmax func-

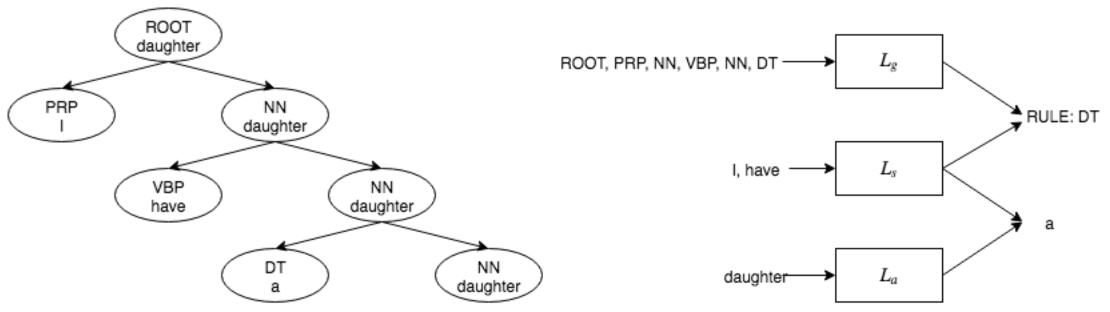

Figure 2: Demonstration of inputs and outputs at node DT. The sequence of inputs to each encoder are shown in the graph. The inputs to L_g is a sequence of labels, rules of parents, and tree depths (only labels are shown). L_s and L_a are used for predicting the word for DT. L_s and L_g are used for predicting the rule for DT. "RULE: DT" indicates DT will be a leaf node since the number of symbols is 1. In this tree, words are generated in the order: daughter - I - have - a.

tion by σ. $E_w \in \mathbb{R}^{|W| \times d_w}$ and $E_r \in \mathbb{R}^{|P| \times d_r}$ are embedding matrices for words, labels, and rules. $A_w \in \mathbb{R}^{n_w \times d_w}$ and $A_r \in \mathbb{R}^{n_p \times d_r}$ are weight matrices (n_w, n_p are the dimensions of input neurons). We use weight tying (Press and Wolf, 2017) to limit the search space for parameters.

Word prediction. To decode for w_k, we use the the hidden states of surface decoder L_s and ancestor decoder L_a. If v_k is a heir, then

$$P(w_k \mid n_k, H_k) = \begin{cases} 1 & w_k = w_{p(k)} \\ 0 & w_k \neq w_{p(k)} \end{cases}$$

Otherwise, the probability of w_k is given by:

$$P(w_k \mid n_k, H_k) = \\ \sigma(tanh([h_s(k); h_a(k); c(k)]\, A_w) E_w^T)$$

At decoding time, we impose an additional constraint that w_k be a valid word for label n_k, to enforce grammaticality. This is estimated from the co-occurrence of w_k and n_k in the tagged training corpus. We only use those words whose frequency of co-occurrence with the given label is above a certain threshold.

Rule prediction. The probability of r_k is given by

$$P(r_k \mid n_k, H_k) = \\ \sigma(tanh([h_s(k); h_g(k); c(k)]\, A_r) E_r^T)$$

Given the definition of L-PCFG syntax tree, we only consider rules that contain n_k at decoding time. There is one exception: at "ROOT" node, only unary rules are considered, and they do not have to contain the label "ROOT".

Hence, we train our architecture by minimizing the negative log-likelihood of words and rules:

$$-\log P(T) = \\ -\frac{1}{|W(T)|} \sum_{\substack{k \\ v_k \neq f(v_{p(k)}, i_{p(k)})}} \log P(w_k \mid n_k, H_k) \\ -\frac{1}{|T|} \sum_k \log P(r_k \mid n_k, H_k)$$

where $|W(T)|$ is the number of non-heir nodes (or the number of words in the original sentence), and $|T|$ is the number of nodes in T. Note that the log probability of each word in the sentence appears exactly once in the above equation. At test time, we conduct beam search and use the same equation to score each generation for selecting words and rules.

In our experiments, we use unlexicalized PCFG as an additional baseline. We still replace the labels of each node with their heirs' in the parse tree returned from Stanford parser, but words are generated only at leaf nodes. This baseline has syntactic structure while generating words from left to right. We use it as a test against top-down generation of words with syntax.

4.7 Training Details

All models are implemented using PyTorch. The hidden size of all LSTM encoders and decoders are 512. The size of embeddings of words, labels, rules, and tree depth are 300. We trained our models using stochastic gradient descent (SGD) with momentum and exponential learning rate decay. Dropout is applied to the input and output layer of LSTMs.

	Stand.	Depend.	Content
Nouns	8.21	7.70	6.25
Verbs	9.56	7.07	7.03
Adjectives	7.38	7.82	7.77
Adverbs	6.38	6.99	7.29
Other	5.64	6.23	6.62

Table 1: Average absolute positions of different type of words.

	Stand.	Depend.	Freq.
1	.0262	.0095	.0002
2	.0149	.0274	.0238
3	.0147	.0116	.0156
4	.0133	.0145	.0147
5	.0134	.0131	.0146

Table 2: Average frequency of first five words in different generation orders.

5 Experiments and Analysis

5.1 Data

We evaluate our model for dialog response generation on Persona dataset ((Zhang et al., 2018)). Each person is give a list of persona descriptions in simple sentences, and they are required to converse according to the given persona. We use last 3 utterances for each response as source. We prepend persona descriptions to source. We use global attention over persona descriptions to compute context vectors. During pre-processing, we truncate all trailing punctuations.

5.2 Positional Statistics

We measure how early do each type of words appear in different generation orders – standard left-to-right order, dependency-based lexicalization (as in Stanford parser), and content-based lexicalization. The earlier a word appear, the less context there is for predicting it. As shown in Table 1, content-based lexicalization can make nouns and verbs appear much earlier, while delaying function words.

To verify frequency-based lexicalization is making infrequent words appear earlier, we show the average frequencies of the first five words. The first few words are more important since they decide the context for generating the following words. In Table 2, the first two words of each parse tree under frequency-based lexicalization are much less frequent.

	Seq2seq	Ours
Standard	3.682	N/A
Dependency	4.015	3.964
Content	4.115	3.865
Frequency	4.088	3.827

Table 3: Perplexities.

5.3 Perplexity

We compare per word likelihood given different generation orders and architectures. For left-to-right sequence decoder, the non-standard generation orders are obtained by linearizing L-PCFG parse trees in pre-order traversal, and words of heirs are not repeated in the linearization. Note that our word model generates word without using rules and labels as inputs to its networks. As can be seen from Table 3, alternative word generation orders all make it harder for standard left-to-right sequence decoder to learn to predict the next word. On the other hand, using a doubly-recurrent architecture, specifically the surface decoder and the ancestor decoder, can improve perplexity scores for top-down word generation over the left-to-right decoder. While our word model with top-down word generation orders has higher perplexity scores than simple model with standard generation order, we emphasize that perplexity is not an appropriate measure for generation tasks.

5.4 Evaluation Metrics

Word Overlap Based Metrics. We use BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and METEOR (Banerjee and Lavie, 2005) scores as automatic evaluation metrics. While the reliability of such metrics has been criticized (Liu et al., 2016), there is also evidence that for task-oriented domains, these metrics correlate with human judgment to a certain extent (Sharma et al., 2017).

Word Embedding Based Metrics. We evaluate the semantic similarity between generated responses and human responses/persona by the cosine distance of their sentence embeddings. We use the word averaging approach by (Arora et al., 2016) to embed the responses, which has been demonstrated to be very good at capturing lexical level semantic similarity. The normalizing singular vector is obtained from the responses in training set.

	Human	Baseline	PCFG	L-PCFG (Dependency)	L-PCFG (Content)	L-PCFG (Frequency)
Length	10.35	7.21	9.39	8.96	**11.04**	**9.74**
BLEU	N/A	0.1926	0.2041	0.2038	**0.2093**	0.2028
ROUGE-L	N/A	**0.1639**	0.1448	0.1571	0.1565	**0.1624**
METEOR	N/A	0.0718	0.0704	0.0721	**0.0777**	0.0739
Cos. Sim. to targets	N/A	**0.0913**	0.0385	0.0824	**0.0964**	0.0883
Cos. Sim. to last utterance	0.1108	0.0880	0.0714	0.0921	**0.1113**	0.0971
Cos. Sim. to persona	0.1489	0.0206	0.0455	**0.0796**	0.0691	0.0605
Distinct uni-gram	6327	678	725	**891**	813	**874**
Distinct bi-gram	44376	2802	3368	**5306**	**5427**	3680
Distinct tri-gram	77884	4844	6061	**10348**	**10787**	6441
Inertia	11771	4385	5027	**6319**	3756	3959
BLEU to training set	0.4334	0.8471	**0.5320**	0.5874	**0.5402**	0.6331
ROUGE to training set	0.4728	0.8970	**0.5837**	0.6701	0.6226	0.7175

Table 4: Evaluation results on Persona dataset.

Context	Human	Seq2seq	L-PCFG
i am great . i just got back from the club	this is my favorite time of the year season wise	that is cool what do you do for a living	awesome ! i am getting ready to go to the club
sure i like tv , what do you watch ?	really anything , what about you ?	i watch a lot of tv	i watch lot of tv movies . i like to watch tv
oh . tell me something about yourself .	well i do not like heights very much and i love animals . what about you ?	i am an accountant . what do you do	i am trying to learn how to work with animals
i live in texas . i love riding my bike here .	are you a christian ? i am jewish	i have never been to the west coast . do you have any hobbies	i do too ! i wish i was there so i can do that for school

Table 5: Examples of generated responses.

Novelty and diversity. We measure word overlapping between generated responses and the responses in training set using BLEU and ROUGE as a proxy for novelty. The responses in training set with most common words with generated responses are used as references. For diversity, we count the number of distinct n-grams. In addition, we perform a k-means clustering on the sentence embeddings of responses into 10 clusters, and measure average squared Euclidean distance between members of each cluster (Inertia). The larger the number, the harder it is to separate embeddings into 10 clusters, thus the greater the diversity.

5.5 Main Results

5.5.1 Quantitative Analysis

Evaluation results are shown in Table 4. For metrics that are not measured using ground truth response as reference, we consider the closer to the number for human responses the better. We first look at measures for overall generation quality. We can see modelling syntactic structures is capable of generating longer responses. BLEU scores are positively correlated with lengths. While syntactic models do better on BLEU, and slightly better on METEOR than sequence-to-sequence baseline, they are generally not on par with the baseline in terms of ROUGE-L, except for frequency-based lexicalization. Among grammar models, lexicalized grammars out-performed unlexicalized grammar.

Relevance is measured using cosine similarities with the previous utterance and persona. Syntactic models with lexicalized grammar beat the baseline in terms of relevance. Furthermore, content-based lexicalization is much more on topic with the last source utterance than dependency-based

	$L_g + L_s$	$L_g + L_s + L_a$	L_g
Lengths	11.04	7.95	8.46
Distinct uni-gram	813	383	376
Distinct bi-gram	5427	1763	1775
Distinct tri-gram	10787	3061	3522
Inertia	3756	2049	2346
BLEU on training	0.5402	0.8056	0.6612
ROUGE on training	0.6226	0.8543	0.7417

Table 6: Ablation studies.

and frequence-based lexicalization. Dependency-based lexicalization is best at being adherent to personas than the other two lexicalization schemes.

All syntactic models generate more novel responses than sequence-to-sequence baseline, as reflected in the last two rows in 4. This is consistent with the observation that sequence-to-sequence model exhibits retrieval-like behaviour, selecting what is most common in the training corpus. Syntactic models also have larger vocabularies. As for cluster analysis, unlexicalized grammar model and dependency-based lexicalized grammar model have better diversity than sequence-to-sequence model; content-based and frequency-based lexicalization have slightly smaller inertia than the baseline.

5.5.2 Qualitative Analysis

We present a few examples generated by sequence-to-sequence baseline and L-PCFG model. There is a clear difference of how left-to-right decoder and L-PCFG tree decoder do conjunctions. Most of the time, standard LSTM decoder combine sentences with periods, while tree decoders learn to use conjunction words, or even clauses.

We also performed an error analysis on the generated responses by L-PCFG, and in Table 7 we selected the most peculiar and representative. These examples are all syntactically fine, but they do not follow the convention of the language or common sense. The first example contains the most common errors in the responses generated by L-PCFG: misuse of prepositions and determiners. It can be fixed by replacing “as a” with “for”. The error of other two examples have even less to do with syntax. The second one misuses the verb “be”, which is probably caused by the high frequency of the word in the corpus. The error of the third example is beyond surface level. Note that phrases such as “cooking as a dinner” and “be a dog” never appear in the corpus. It is clear that L-PCFG models are learning to make compositions of words and phrases, unlike standard LSTM decoder, which seems to only memorize word combinations.

i am doing well . just finished cooking as a dinner
i am sure it is nice . i am going to be a dog
i like to ride my horses on my bike

Table 7: Some peculiar examples generated by L-PCFG models.

5.6 Ablation Studies

We perform ablation studies on our architecture, with content-based lexicalization. Specifically, we consider two alternative ways of making rule prediciton. The first one takes only the hidden state of L_g for predicting rules, hence making the prediction of rules entirely independent from words. The second one takes both the hidden states of L_s and L_a, together with L_g, for predicting rules, in which way future lexical information is used for the construction of syntax trees.

For rule prediction using the hidden states of both surface decoder and ancestor decoder, we noticed a significant drop in diversity in generated responses. The model over-predicts “what do you do for a living” for than 50% of the time, and the lengths of responses tend to be shorter.

For the other choice in which rule prediction is independent from words, the results are closer to the original model, but there are still some decreases in lengths and lexical diversity. Upon manual inspection, we found that this model behaved more like sequence-to-sequence model. There are less compound sentences, and more conjunctions of simple sentences by end punctuation marks.

The proportion of simple sentences are also larger.

6 Conclusion

We consider the problem of generating natural language in alternative orders and with syntactic tree structures, with the use of lexicalized grammar. By incorporating syntactic structures, our models are capable of generating longer sentences. By changing lexicalizaion schemes and making content words appear earlier in generation process, our models are able to make word choices that are more relevant to source. Furthermore, incorporating syntax facilitates response generation with richer vocabularies and more complex structures. On the other hand, as shown in our error analysis, there is still room for improvement on discourse and pragmatics level.

Acknowledgments

This material is based upon work partially supported by the National Science Foundation (Award # 1722822). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of National Science Foundation, and no official endorsement should be inferred.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 132–140.

David Alvarez-Melis and Tommi S Jaakkola. 2016. Tree-structured decoding with doubly-recurrent neural networks.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Gary S Dell, Franklin Chang, and Zenzi M Griffin. 1999. Connectionist models of language production: Lexical access and grammatical encoding. *Cognitive Science*, 23(4):517–542.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 33–43.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 72–78.

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George Dahl. 2018. The importance of generation order in language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2942–2946.

Jetic Gū, Hassan S Shavarani, and Anoop Sarkar. 2018. Top-down tree structured decoding with syntactic connections for neural machine translation and parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 401–413.

Dan Klein and Christopher D Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, pages 3–10.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

Jingyuan Li and Xiao Sun. 2018. A syntactically constrained bidirectional-asynchronous approach for emotional conversation generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 678–683.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132. Association for Computational Linguistics.

Shikib Mehri and Leonid Sigal. 2018. Middle-out decoding. In *Advances in Neural Information Processing Systems*, pages 5519–5530.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. 2018. A tree-based decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4772–4777.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 698–707.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 440–450.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213. Association for Computational Linguistics.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 310–320.

Ganbin Zhou, Ping Luo, Rongyu Cao, Yijun Xiao, Fen Lin, Bo Chen, and Qing He. 2018. Tree-structured neural machine for linguistics-aware sentence generation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5722–5729. AAAI Press.

A Appendices

A.1 Proof sketch for *Proposition* in Section 4.4

We prove by contradiction. Suppose there is a node n_k, whose lexicalization is not in the leaves before node k, nor in the ancestors of node n_k. There must be a leaf l inheriting the lexicalization of node n_k. Since n_k is not an ancestor of node k, l must be a leaf before k, so we have a contradiction.